LavA: An Embedded Operating System for the Manycore Age
Michael Engel, Matthias Meier and Olaf Spinczyk
Embedded System Software, Technische Universität Dortmund

Overview

Manycore systems are a growing and inevitable trend in embedded systems. An open question is how to use the abundant computing resources to improve typical properties of embedded systems, like real-time constraints, energy consumption and speed of computation.

The LavA approach proposes new operating system structures that permit for easier construction and analysis of application-specific embedded systems.

Project Goals

New OS Structures for Embedded Manycore Devices

Reducing Operating System Overhead
- Implementing parts of the OS in hardware
- Static assignment of processes to cores

System Software and Hardware Co-configuration
- Application-driven configuration of the OS and the hardware

Improvement of Real-time Properties

Reduction of Energy Consumption
- Dynamic and separate frequency adjustment for each core

Hardware Platform

The hardware platform for LavA is a configurable and scalable manycore system with lots of parameters to adapt the system to application needs.

Characteristics of the hardware platform
- Different cores can be used depending on application requirements
- Local memory for each core
- Integration of specialized IPs (e.g., FFT Unit)
- Flexible communication structures for IPC
- Selection of several peripherals (e.g., CAN or UART Controller)

Manycore OS Structure

Typical operating systems for small embedded devices use a fixed set of tasks. Such structures can be easily mapped to our hardware platform. When sufficient chip space is available, we can offer one CPU core for each task.

This has major effects on the required operating system functionality:
- CPU scheduling is unnecessary
- No memory protection is required
- Methods for communication and synchronization of tasks required
- Drivers for peripheral devices are required

This reduces the overhead, since the operation system does not have to care about context switches.

WCET [3] calculation tools can be used to estimate a lower bound for a core’s frequency to save energy. Furthermore, we can stop a waiting core completely until the arrival of e.g., an interrupt, when having a single task per core.

To match the real-time constraints is a lot easier when only a single task is executed on each CPU.

Design Flow

Literature